Review of Inputs

- Geocoded points
  - What is the surrogate method to be used
    - This is important as most of the high cost customers have un-geocoded addresses.
    - When should the surrogate method be used
      - May need to use all surrogate points if geocoding success is less than xx%
  - Do we include housing units
- Does not appear to be able to use actual data
  - Loop counts by office (FCC lines File) (FCC Criteria 1)
  - Wire Center Boundaries and locations

Review of Code

- Use of Turbo Pascal and current coding structure
  - Not a common language
  - Difficult to Test
  - Difficult to Audit/Verify
  - Difficult to Review
- We have converted all HCPM Turbo Pascal to pseudo code
  - Somewhat easier to review
  - Generic Syntax
  - Can use to convert system to a more universal language: VB

Review of Code (cnt'd)

- Programming Review
  - Numerous variables were not used
    - Duct_cost_per_kf, Copper_line_max, T1_line_max, Th2016, Th672, Th96, SpclAccessLines_per_bus, CriticalWaterDepth, WaterFactor, SoilTexFactor
  - In Structur_Cost_Fn, there is an excess amount of Looping.
    - A lot of the loops could be eliminated if a variable for the density index was created.
    - A lot of code could be elminated if a set of factors was assigned
  - The SurfaceText array should be sorted and binary serach used.
    - This is a fairly large array and is now linearly searched
  - The performance of most lookups/searches could be improved if the lookups were exited when a match is found rather than continuing on to the end of the loop

Review of Code (cnt'd)

- Programming Review
  - It would appear that multiple occurrences of the minimum spanning tree could be eliminated.
    - The functions are globally defined and then basically overridden by local instances of these functions
  - Use of a lot of Global Variables should be avoided
  - Passing Global variables as parameters to procedures or functions should be avoided
  - It appears that the code is a combination of several modeling efforts
    - Coding style suffers as there is no consistency
  - The Logic is extremely difficult to verify
    - Need to create audit trail to assist the understanding of the code
    - Need to improve documentation of code

Review of Code (cnt'd)

- Logic Comments
  - Cable sizes for feeder and distribution are consistently undersized. See Function Feed_Cable_Cost and Function Dist_Cable_Cost. It appears that the cable sizing lookups are reversed. The cable that is smaller than the number of lines is chosen instead of the next bigger size cable.
  - The factors for soft_rock structure and normal structure are reversed. See Function Structur_Cost_Fn. Looks like the values in the SoilTexture file are used backwards.
  - The cumulative density factor is understated. The density for each entity is calculated correctly. However, then the numbers are cumulated the calculated density is used instead of the cumulative area divided by the cumulative lines.
  - The cost of 24 gauge copper is assumed to be a constant multiplier of 26 gauge copper costs. The 24_ multiplier is used even though there is an input file of 24 gauge copper costs.
  - The copper capacity factors are used to size the fiber cables for feeder cable.
  - The file fdrmix.txt is used to populate both the CopFeedPlantMix array, and the FibFeedPlantMix array. If these arrays are meant to be identical, then only one of the arrays should be used. If they are separate because of the possibility that they might contain different data, then separate txt files should be used to populate them.

Review of Code (cnt'd)

- Logic Comments
  - The file 26g.txt is used to populate both the CopDistCost array, and the CopFeedCost array. If these arrays are meant to be identical, then only one of the arrays should be used. If they are separate because of the possibility that they might contain different data, then separate txt files should be used to populate them.
  - It looks like the long loop penalty is applied at least twice.
  - Not sure DS1 costs are properly calculated. The units carried in the record are neither lines nor channels. Also the original input data is modified, generally you should try to avoid this.
  - Not sure the distance associated with linking cables are carried forward.
  - Version 2.6 added the variable PrimCutOffDensity. This is used in distrib.pas in the conditional expression:

```
(v2.5)  if UsePrimDist or (density < 100) then ..
(v2.6)  if UsePrimDist or (density < PrimCutOffDensity) then
  ..
```

PrimCutOfDensity is set to 0.

Review of Engineering

- Use of outdated T-1
  - T-1 technology was used in the 60's and 70's, used today only to reinforce existing copper runs
  - Fiber is the forward looking technology for long loops
- T-1 costs understated due to lack of repeaters
- Copper Lengths appear to exceed standards
  - BOC Notes refer to total loop length, HCPM uses this for distribution only (violates engineering)
    - it is important to remember that what is engineered for the total loop might not work on the same distance in just the feeder or just the distribution
  - There are known limitations of 26 gauge copper
    - 9kft off of DLC terminal (12kft if mix of 24/26)

Review of Engineering (cnt'd)

- Cable size selection appears to choose the next smaller cable
  - If 2200 lines required, model would pick 1200 pair cable
- If a manhole has more than 9 ducts an incremental cost per duct is added
  - However this cost is apparently not divided by the manhole spacing
- Structure costs seem to be lacking ducts and inner ducts
- Manhole and Pole costs derived on a per foot basis
  - in the network planning, the manholes (poles) are placed at specific intervals
  - each underground (aerial) cable run needs to start and end with a manhole (pole), so the tendency is to forget to place the first or last manhole (pole)
  - applying manhole (pole) costs on a per foot basis increases the chances of underestimating this structure
- There is no manhole/handhole/pullbox investment in distribution
- Splicing Costs appear to be based on a single value, independent of cable size

Review of Engineering (cnt’d)

- Need to gain better understanding of inputs
  - For example, there are no separately defined installation costs
    - Do we assume that input values represent material and placing
    - However, Fiber_splice_cost has been separately identified as a variable
  - What do values in
- DLC Central Office Terminal costs seem to be overlooked
  - Documentation states that the COT line card is included
  - Does not mention COT
    - This can be shared with multiple Remote sites
- DLC System sizes of greater than 1344 are used
  - We know they are available
    - However, they are not standard
    - They are quite large and exceed size limitations for rights-of-way
    - If they are used, other costs may need to be included (e.g., land)

Review of Engineering (cnt'd)

- Concerns with Loop Length Optimization
  - There are concerns about underestimating loop length
    - Need to make sure that the model does not violate FCC Criteria 1, in which the objective is to obtain loop lengths that match incumbent carrier's actual loop lengths
    - Minimum spanning tree algorithms have been proven to underestimate actual network loop lengths
      - Understatement varies in rural and Urban areas
        - » Rural length can be understated by 20-30 percent, urban areas by more than 100%
      - Need to make sure that the model will account for this

Partial Review of Documentation

- Table 2 has values that appear to represent different terminal sizes. The drop termination input file (drop.txt) however has the same value for all terminals (by OSP type) and these values are less than 60% of the smallest values shown in the documentation.
- On page 19 of the documentation, it is stated that " When 24-gauge copper is required we apply a multiplier to these values, with a default value of 1.25". Table 16 in the same document lists the value as 1.1736. The input file (feeddist.prm) however contains a value of 1.0 for this variable.
- Sort order of input files are critical, but the program contains no sorts to insure that the data is in the correct order.
- The documentation does not contain explanations of many of the input values.

Review of Running/Validation

- More state data is needed to validate the model vis-à-vis other models
- Hard to validate against existing models due to
  - use of On-target data
  - Lack of Output
  - Lack of Auditing steps
- Model froze in Windows 97 on some machines
- When able to run, MD took over 11 hours to process
  - When complete, could not easily determine what needed to be viewed
  - Only provides Loop capital costs
    - No subisidy
    - No Company, Parent, or Small-Medium-Large Information
    - No NID, Drop, Terminal or FeedSplice cost
    - MD cost of 7.91 per month

Review of Proxy Modeling

- While the HCPM may provide the basis for an acceptable loop model
  - Loop plant is generally 30% of the ILEC cost for basic service
- How will other parts of the proxy model be addressed:
  - Switch
  - Interoffice
  - Signaling
  - Operating Expenses
  - Capital Costs
  - Support Investments
  - Reporting
  - Subsidy calculation
  - User inputs

DRAFT >>>>>Proxy System Workplan <<<<< DRAFT

Potential National Proxy Model Workplan (based on BCPM workplan):

- The National Proxy modelcould be built around the logic coded by each module team.
- Each EXCEL/VB/TP Logic Module team could work independently to develop the appropriate algorithms, necessary Inputs, Outputs, and procedures
  - Based on accepted specifications
- The Overall System Team work will be broken up into a total of 7 distinct modules. These Modules and the Basic System Schematic are pictured in Figure 1. Each team would be expected to develop the module to fit into the completed system and to develop basic documentation that a reviewer would understand.

- General Notes
  - System response must be reasonable
  - System resource requirements (RAM, Harddisk, processor speed) should be minimized
  - Code and structure should be simple, easy to understand, verifiable, reviewable by non-techies, and modifiable
  - System documentation must be complete and understandable (Pictures are encouraged)
  - General Database Guidelines and Structures should be adhered to somewhat
    - Number of Tables should be minimalized
    - Impact on run time avoided
    - Non-optimized structure is OK in the context of simplifying
  - VB , Macros, and SQL code should be well structured and self documenting

WORK MODULES:

1. User Interface
   - Preliminary Plans are that this will be written in ?? (Visual Basic)
   - This will control the use of the entire model
   - This Module will also control Scenario processing, whereby the user can change inputs, raw files and/ or Module logic and save these results and changes under various Scenario ID's
     - The Scenario should keep record of User inputs, raw and Logic
     - In addition, this Scenario Information should be passed to the reporting Module
2. Raw GIS data
   - These may exist as either CSV flat files by state or in an Access Database
   - This is the base data for the model
   - The base values are not user adjustable. However, the user can substitute their own raw files in a scenario analysis
   - There will probably be three Raw files
     - Housing Unit, Household and Business line counts
     - Cluster to CBG conversion File
     - Terrain Data by Cluster
     - CLLI information File
       - Boundary
       - Lat, and Long
       - Company Ownership

            - Parent Company
            - Large, Medium or Small

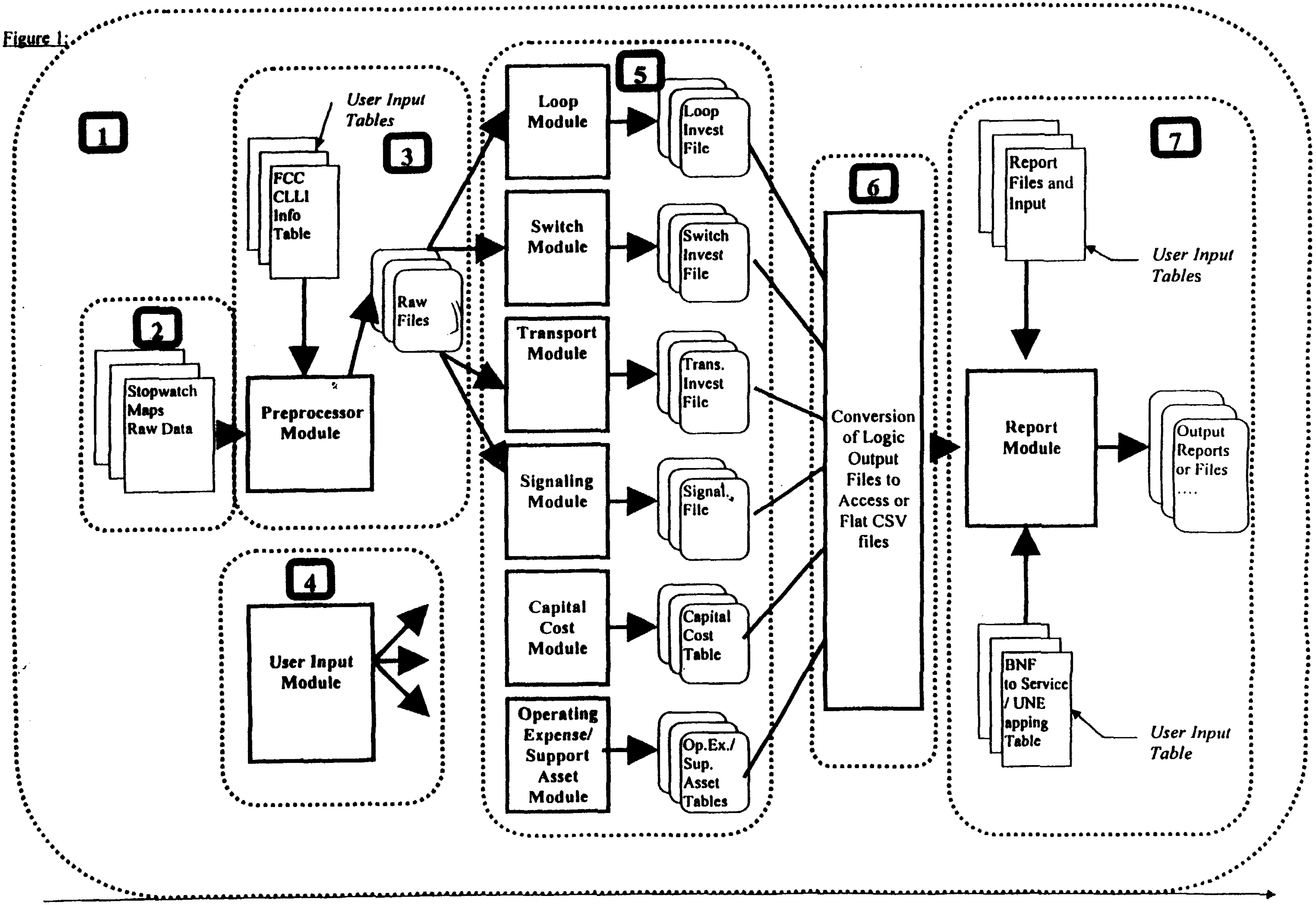
3. Preprocessor Module
    - Written in either VB5, EXCEL, or ACCESS
    - It will combine the FCC CLLI file Line data with the GIS files
        - This will create the Line counts by Cluster by grid CSV files
            - Logic will need to be created to True the Grid Household and Business line data so that when summed they match the values in the FCC CLLI file
                - Additional logic will need to be written to account for the fact that some FCC CLLI data may not exists for some CLLI. Therefore, we will need Global Defaults for adjustments
    - This will create the various CSV files needed by the Engineering logic modules
        - The Loop module will need the Combined FCC/GRID file at the GRID level
        - The Switch/Transport/Signaling modules will need a Summarized FCC/GRID file at the CLLI level
    - This will also create any files needed by the Other Module teams
4. User Input Data Module
    - Possibly written in VB5/Excel
    - This will pull out all of the User Adjustable data from the Engineering Logic Modules into a common area
    - Logical Setup, wifh some simple data edit checks
5. Engineering Logic Module
    - Written in ??
    - As previously mentioned, each Engineering Logic Work team could develop their own Logic
    - These teams will
        - Develop the input and output routines
        - Standardize the look and structure of the logic
6. Conversion Module
    - Developed in either Access or CSV format
    - This will standardize the input file structure from each Engineering Logic module into a Database structure
7. Report Module
    - Developed in Either Access or VB/Excel
    - This will control the logic of reports
    - User will input various Report Variables
        - Type of Report
        - Level of Report
            - State
            - Company
            - etc..
        - Variables for Reporting
            - Benchmarks
            - Investment Cap
            - Level of Subsidy
                - CBG

DRAFT >>>>>Proxy System Workplan <<<<< DRAFT

        - Grid
        - CLLI
- The reporting module will develop the logic to
    - Combine the Engineering Logic output files
    - Use the Reporting Variables

DRAFT >>>>>Proxy System Workplan <<<<< DRAFT

Figure 1:



What is next

- Continuing review
  - Pseudo Coding CLUSTINTF
  - Engineering review of FeedDist
  - Review of outputs
- We can convert the HCPM to VB
  - More supportable
  - Can create audit steps
- Need to begin work on other parts of the model

HCPM
Version 2.6

Pseudo code for FeedDist

as prepared by the BCPM Sponsors

:PM version 2.6 Pseudo code

:face
ı document was created by taking the HCPM Pascal source code and converting it into pseudo code. The pseudo code was designed so ! the details of the calculations performed in the code were retained, while the other aspects of the source code such as file handling I memory management were summarized or removed.

rsion 2.6
ı document was originally created using HCPM version 2.5. It has been modified to include the changes from version 2.6. These nges are highlighted. The following modules have changes: global, feeddist, distrib, feeder, primdist and primfeed.

claimer
ile every effort was made to generate pseudo code that accurately reflects the logic in the Pascal code, it is recommended to refer to Pascal code if there are any questions.

ıwsing The Document

: document is divided into sections - each Pascal module is in its own section. To browse by section, use the buttons found at the tom of the right-hand scroll bar. Press the circle button in the middle and select "Browse By Section". Then press the double arrow tons to browse up or down by sections.

ble Of Contents
: table of contents has links to the individual modules and the functions and procedures within each module. The functions and cedures that are displayed in red are local to the module and are not called from code outside of that module.

get to the Table Of Contents at any time, press [CTL+T]. (If this doesn't work, it may be because you have opened the document with cros disabled.)

rmatting
/les were used in formatting the text. The styles can be edited so that the text will stand out in a printed version of this document.

ocedure and function names are formatted with the style ProcOrFunction.

obal variable names are formatted with the style *Global Variable*.

ırnings are formatted with the style WarningMsg.

mments are formatted with the style *Comment*.

e original comments from the source code are in curly brackets {}

r explanatory text that replaces code, the text will appear as normal text.

Procedure Argument Syntax
Variables that are passed to a procedure (including functions) can also return values set by the called procedure. Such variables are proceeded with a *, both in the procedure definition, and in the actual call to the procedure.

'able of Contents

globals.pas

lobals.pas

lobal constants:

ero = 0
ne = 1
alf = .5

lobal variables:

Variable Name	Procedure Where Value Is Set
R	process (feeddist.pas)
R	process (feeddist.pas)
A	process (feeddist.pas)
oordinate file	process (feeddist.pas)
ridfile	process (feeddist.pas)
[illegible]file	process (feeddist.pas)
utfile	
reaname	feeddist.pas - main program
ax drop length	get user parameters (global.pas)
ser lambda	get user parameters (global.pas)
ost per drop kf	get user parameters (global.pas)
eed copper cable capacity	get user parameters (global.pas)
ist copper cable capacity	get user parameters (global.pas)
id cost	get user parameters (global.pas)
uct cost per kf	get user parameters (global.pas)
akerate	get state data (global.pas)
ines per bus	get user parameters (global.pas)
ines per house	get state data (global.pas)
ax copper distance	get user parameters (global.pas)
axCopperPenalty	get user parameters (global.pas)
opper gauge mover	get user parameters (global.pas)
opper t1 mover	get user parameters (global.pas)
1 fiber mover	get user parameters (global.pas)
opper line max	get user parameters (global.pas)
1 line max	get user parameters (global.pas)
iber cable capacity	get user parameters (global.pas)
h2016	get user parameters (global.pas)
h672	get user parameters (global.pas)
h96	get user parameters (global.pas)
2016	get user parameters (global.pas)
2016	get user parameters (global.pas)
672	get user parameters (global.pas)
672	get user parameters (global.pas)
96	get user parameters (global.pas)
96	get user parameters (global.pas)
24	get user parameters (global.pas)
24	get user parameters (global.pas)
c96	get user parameters (global.pas)
c96	get user parameters (global.pas)
c24	get user parameters (global.pas)
c24	get user parameters (global.pas)

globals.pas

Variable Name	Procedure Where Value Is Set
SA array	process (feeddist.pas)
i	
NumDensZones	get user parameters (global.pas)
NumCableSizes	get user parameters (global.pas)
NumFeedCableSizes	get user parameters (global.pas)
NumFiberCableSizes	get user parameters (global.pas)
NumDropTerminalSizes	get user parameters (global.pas)
NumManholeSizes	get user parameters (global.pas)
NumXBoxSizes	get user parameters (global.pas)
NumTaxTypes	get user parameters (global.pas)
CopDistCost[]	get user parameters (global.pas)
DropTermCost[]	get user parameters (global.pas)
CopFeedCost[]	get user parameters (global.pas)
FiberFeedCost[]	get user parameters (global.pas)
IntfcCost[]	get user parameters (global.pas)
NormalStruc[]	get user parameters (global.pas)
SoftRockStruc[]	get user parameters (global.pas)
HardRockStruc[]	get user parameters (global.pas)
ManholeCost[]	get user parameters (global.pas)
ManholeSpec[]	get user parameters (global.pas)
DistPlantMix[]	get user parameters (global.pas)
CopFeedPlantMix[]	get user parameters (global.pas)
FibFeedPlantMix[]	get user parameters (global.pas)
FillFact[]	get user parameters (global.pas)
SurfText[]	get user parameters (global.pas)
Sharing[]	get user parameters (global.pas)
t1 redundancy factor	get user parameters (global.pas)
pct ds1	get user parameters (global.pas)
pct isa	get user parameters (global.pas)
SpclAccessRatio	get state data (global.pas)
SpclAccesslines per bus	get user parameters (global.pas)
multiplier 24	get user parameters (global.pas)
copper placement depth	get user parameters (global.pas)
fiber placement depth	get user parameters (global.pas)
CriticalWaterDepth	get user parameters (global.pas)
WaterFactor	get user parameters (global.pas)
MinSlopeTrigger	get user parameters (global.pas)
MinSlopeFactor	get user parameters (global.pas)
MaxSlopeTrigger	get user parameters (global.pas)
MaxSlopeFactor	get user parameters (global.pas)
CombSlopeFactor	get user parameters (global.pas)
SoilTaxFactor	get user parameters (global.pas)
DistRoadFactor	get user parameters (global.pas)
FeederRoadFactor	get user parameters (global.pas)
FiberFillFactor	get user parameters (global.pas)
DistanceType	get user parameters (global.pas)
num SAs	process (feeddist.pas)
mgcounter	calculate microgrid cost (distrib.pas)
tot tterms	process (feeddist.pas)
tot fterms	process (feeddist.pas)
tot reallines	process (feeddist.pas)
tot buslines	process (feeddist.pas)
tot dropfeet	process (feeddist.pas)

globals.pas

Variable Name	Procedure Where Value Is Set
tot distfeet	process (feeddist.pas)
tot feedfeet	process (feeddist.pas)
tot feedcost	process (feeddist.pas)
tot distcost	process (feeddist.pas)
tot dropcost	process (feeddist.pas)
tot ftcost	process (feeddist.pas)
tot ttcost	process (feeddist.pas)
tot fdicost	process (feeddist.pas)
tot midcost	process (feeddist.pas)
tot dtcost	process (feeddist.pas)
tot sais	process (feeddist.pas)
tot SpclAcclines	process (feeddist.pas)
tot households	process (feeddist.pas)
lines served	process (feeddist.pas)
grid lines served	process (feeddist.pas)
Total investment	process (feeddist.pas)
Total lines	process (feeddist.pas)
average investment	process (feeddist.pas)
temp	
avg droplength	process (feeddist.pas)
avg distlength	process (feeddist.pas)
avg feedlength	process (feeddist.pas)
i	
do batch	feeddist.pas - main program
input exists	process (feeddist.pas)
batchfile	feeddist.pas - main program
feed ugd cable	process (feeddist.pas)
feed bur cable	process (feeddist.pas)
feed aer cable	process (feeddist.pas)
feed ugd fiber	process (feeddist.pas)
feed bur fiber	process (feeddist.pas)
feed aer fiber	process (feeddist.pas)
eed ugd structure	process (feeddist.pas)
eed bur structure	process (feeddist.pas)
ead aer structure	process (feeddist.pas)
eedManholeCost	process (feeddist.pas)
alcCuFeedFill	process (feeddist.pas)
alcCuDistFill	process (feeddist.pas)
ot DLC lines	process (feeddist.pas)
ist ugd cable	process (feeddist.pas)
ist bur cable	process (feeddist.pas)
ist aer cable	process (feeddist.pas)
ist ugd structure	process (feeddist.pas)
ist bur structure	process (feeddist.pas)
ist aer structure	process (feeddist.pas)
istManholeCost	process (feeddist.pas)
ugd cap	get user parameters (global.pas)
bur cap	get user parameters (global.pas)
aer cap	get user parameters (global.pas)
ugd fib	get user parameters (global.pas)
bur fib	get user parameters (global.pas)
aer fib	get user parameters (global.pas)
ugd struc	get user parameters (global.pas)

globals.pas

Variable Name	Procedure Where Value Is Set
ac bur struc	get user parameters (global.pas)
ac aer struc	get user parameters (global.pas)
ac tl term	get user parameters (global.pas)
ac fib term	get user parameters (global.pas)
ac fdi	get user parameters (global.pas)
ac manhole	get user parameters (global.pas)
CloseWindow	feeddist.pas - main program
VerboseOut	feeddist.pas - main program
UsePrimDist	feeddist.pas - main program
CmdParam	feeddist.pas - main program
tot area	process (feeddist.pas)
PrimCutOffDensity	feeddist.pas - main program
code	feeddist.pas - main program
state	get state data (global.pas)
this state	get state data (global.pas)
splice cost	get cost data (global.pas)
ac splice	get cost data (global.pas)
feed splice cost	process (feeddist.pas)
ac drop	get cost data (global.pas)
ac drop term	get cost data (global.pas)
ac mid	get cost data (global.pas)
max BAfe	get user parameters (global.pas)

```
function max

    passed variables:
    x
    y

    //pass two variables, return the larger of the two

    if x >= y then
        max = x
    else
        max = y
    end if


function min

    passed variables:
    x
    y

    //pass two variables, return the smaller of the two

    if x <= y then
        min = x
    else
        min = y
    end if
```

```
 ion sqr

 passed variables
 x

 //pass a number, return the square of the number

 sqr = x * x



tion fill_factor_fn

 passed variables:
 density
 feeder_indicator

 local variables:
 i
 temp


 //Loop from 1 to the value in NumDensZones using i. For each value of i, index the FillFact array to get the values for density,
 FeedFillFactor, and DistFillFactor, as appropriate.


 for i = 1 to NumDensZones
      if density >= FillFact(i).density then
           if feeder_indicator = 1 then
              temp = FillFact(i).FeedFillFactor
           else
              temp = FillFact(i).DistFillFactor
           end if
      end if
 next

 fill_factor_fn = temp



cedure get_user_parameters

 local variables:
 infile

 read the following variables from the file FEEDDIST.PRM (user parameters):

 max_drop_length
 user_lambda
 takerate
 lines_per_house
 copper_gauge_xover
 multiplier_24
 max_copper_distance
 MaxCopperPenalty
 copper_t1_xover
```

```
      t1_fiber_xover
      copper_line_max
      t1_line_max
      t1_redundancy_factor
      feed_copper_cable_capacity
      dist_copper_cable_capacity
      fiber_cable_capacity
      copper_placement_depth
      fiber_placement_depth
      CriticalWaterDepth
      WaterFactor
      MinSlopeTrigger
      MinSlopeFactor
      MaxSlopeTrigger
      MaxSlopeFactor
      CombSlopeFactor
      SoilTexFactor
      th2016
      th672
      th96
      pct_del
      pct_lea
      SpclAccessRatio
      lines_per_bus
      SpclAccessLines_per_bus
      DistRoadFactor
      FiberFillFactor
      DistanceType
      FeederRoadFactor
      max_RATs


Procedure get_cost_data
      local variables:
      infile
      i

      read the following variables from the file FDCOST.TXT (cost data):

      cost_per_drop_kf
      nid_cost
      duct_cost_per_kf
      a2016
      b2016
      a672
      b672
      a96
      b96
      a24
      b24
      ac96
      bc96
      ac24
      bc24
      splice_cost

      read the following variables from the file ANNCHG.TXT (annual charge data):
```

```
ac_ugd_cop
ac_bur_cop
ac_aer_cop
ac_ugd_fib
ac_bur_fib
ac_aer_fib
ac_ugd_struc
ac_bur_struc
ac_aer_struc
ac_manhole
ac_t1_term
ac_fib_term
ac_fdi
ac_splice
ac_drop
ac_drop_term
ac_nid
```

read the values for the array CopDistCost from the file 26g.txt:

```
NumCableSizes = 0

for each line in the file
     NumCableSizes = NumCableSizes + 1
     read in
       CopDistCost[NumCableSizes].CableSize
       CopDistCost[NumCableSizes].CostUgd
       CopDistCost[NumCableSizes].CostBur
       CopDistCost[NumCableSizes].CostAer
```

read the values for the array DropTermCost from the file drop.txt:

```
NumDropTerminalSizes = 0

for each line in the file
     NumDropTerminalSizes = NumDropTerminalSizes + 1
     read in
       DropTermCost[NumDropTerminalSizes].size
       DropTermCost[NumDropTerminalSizes].CostBur
       DropTermCost[NumDropTerminalSizes].CostAer
       DropTermCost[NumDropTerminalSizes].CostUgd
```

read the values for the array CopFeedCost from the file 26g.txt:

```
NumFeedCableSizes = 0

for each line in the file
     NumFeedCableSizes = NumFeedCableSizes + 1
     read in
       CopFeedCost[NumFeedCableSizes].size
       CopFeedCost[NumFeedCableSizes].CostUgd
       CopFeedCost[NumFeedCableSizes].CostBur
       CopFeedCost[NumFeedCableSizes].CostAer
```

*W – The file 26g.txt is used to populate both the CopDistCost array, and the CopFeedCost array. If these arrays are meant to be identical, then only one of the arrays should be used. If they are separate because of the possibility that they might contain different data, then separate txt files should be used to populate them. The way it is now, they will always be identical.

read values for the array FiberFeedCost from the file fibrcabl.txt:

```
NumFiberCableSizes = 0

for each line in the file
     NumFiberCableSizes = NumFiberCableSizes + 1
     read in
       FiberFeedCost[NumFiberCableSizes].size
       FiberFeedCost[NumFiberCableSizes].CostUgd
       FiberFeedCost[NumFiberCableSizes].CostBur
       FiberFeedCost[NumFiberCableSizes].CostAer
```

read values for the array IntfcCost from the file fdi.txt:

```
NumXCBoxSizes = 0

for each line in the file
     NumXCBoxSizes = NumXCBoxSizes + 1
     read in
       IntfcCost[NumXCBoxSizes].NumLines
       IntfcCost[NumXCBoxSizes].Cost
```

read values for the array NormalStruc from the file normal.txt:

```
NumDensZones = 0

for each line in the file
     NumDensZones = NumDensZones + 1
     read in
       NormalStruc[NumDensZones].Density
       NormalStruc[NumDensZones].FeedUgd
       NormalStruc[NumDensZones].DistUgd
       NormalStruc[NumDensZones].FeedBur
       NormalStruc[NumDensZones].DistBur
       NormalStruc[NumDensZones].FeedAer
       NormalStruc[NumDensZones].DistAer
```

read in the values for the array SoftRockStruc from the file softrock.txt:

```
for i = 1 to NumDensZones
     read in
       SoftRockStruc[i].Density
       SoftRockStruc[i].FeedUgd
       SoftRockStruc[i].DistUgd
       SoftRockStruc[i].FeedBur
       SoftRockStruc[i].DistBur
       SoftRockStruc[i].FeedAer
```